Attorney Docket No.: 021202-000200US
Client Reference No.: QST-026US

PATENT APPLICATION

PREDICTIVE RESOURCE ALLOCATION IN COMPUTING SYSTEMS

Inventor: Ian Hirschsohn, a citizen of The United States, residing at
249 So. Hwy. 101, #270
Solana Beach, CA 92075

Assignee: QuickSilver Technology, Inc.
6640 Via Del Oro, Suite 120
San Jose, CA 95119

Entity: Small

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
Tel: 415-576-0200

PREDICTIVE RESOURCE ALLOCATION IN COMPUTING SYSTEMS

COPYRIGHT NOTICE

BACKGROUND OF THE INVENTION

[02] The present invention relates generally to the field of computer and information processing and more specifically to a system for predictably allocating computing resources within a multi-processor (as opposed to a single processor) computing environment.

[03] Conventional systems for allocating resources are generally known. Resource allocation issues arise in a broad range of industrial and technological areas such as routing of data in communication channels, inventory control and operation management for example. As used herein, resource allocation refers to allocation of memory and other processor resources within a multi-processor computing environment in order to optimize data transfer.

[04] The importance of resource allocation in multi-processor environments cannot be overemphasized. A demand for increased processing power provided by multiple-processor systems currently exists. In parallel computing for example, a plurality of processors are utilized for increased processing power. However, because availability is limited, the plurality of processors must share the limited computing resources. Without resource allocation between the various processors, problems of resource contention and deadlock could arise, virtually shutting down the computing system.

[05] Conventional systems for allocating memory in multi-processor systems are generally known to follow the paradigm in single processor environments. In an IRQ/ISR model for example, resource allocation is reactive. A task requiring a resource must request the resource before it is allocated. By way of example, a program desiring 100k of

memory buffer must typically request the memory from a processor. The request is then granted by the processor in conjunction with a memory controller. Disadvantageously, the processor must stop its current activity to service the request such that a bottleneck for data transfer is created. This static memory allocation scheme may significantly degrade performance in a multi-processor computing environment.

[06] This paradigm makes little or no sense in a multi-processor computing environment where tasks are executed in parallel such that a first process and a second process are run simultaneously by a first and a second processor, respectively. Sometimes the second processor may need to wait for results from the first processor in order to continue processing. At other times however, resource allocation occurs in a manner wherein although the first and second processors are independent of each other, each processor must request and thereafter await receipt of computing resources. Each processor is inhibited from beginning or continuing processing if there are no resources when needed. Only after the computing resources are allocated can further processing continue.

[07] Another disadvantage of conventional systems is that computing resources are typically allocated in advance of execution of the task that requests the computing resource. If memory space is required by the first processor, the memory space is immediately allocated even though it not needed until sometime later. In so far as the memory space is occupied prior to need and after execution, such prior resource allocation is wasteful as the space cannot be allocated to other processes. The memory space is only needed during execution of the program not before or after.

[08] A further disadvantage of conventional multi-processor system is that all of the processors in the system perform resource allocation services. This provides for inefficiency since in addition to other tasks to be performed, each processor is burdened with the responsibility of resource allocation.

[09] Therefore, there is a need to resolve the aforementioned disadvantages specific to multi-processor computing environments and the present invention meets this need.

BRIEF SUMMARY OF THE INVENTION

[10] According to a first aspect of the present invention, a system for allocating computing resources to other processors with a multi-processor computing environment is disclosed. A processor is dedicated to providing resource allocation services to the other processors within the multi-processor computing environment. A script file is provided to the dedicated processor, the script containing information related to the

computing resources required by the other processors. Among other information, the script file is parsed by the dedicated processor to determine the resources required by the other processors and when the resources are required. Thereafter, the dedicated processor dynamically allocates the resources at the time needed by the other processors. Because the script file is provided, multiple processes can be synchronized, and each processor knows ahead of time which tasks can be performed without having to wait for another processor. Not only are resources dynamically allocated, resource allocation is synchronized with the time such resources are needed by the other processors.

[11] According to another aspect of the present invention, in a multi-processor computing environment, a method by a first processor for allocating resources for use by a second processor is disclosed. The method includes: (1) providing a script to the first processor, the script containing information related to the resources required by the second processor; (2) parsing the script to determine the resources required by the second processor; and (3) dynamically allocating the resources at the time needed by the second processor.

[12] According to another aspect of the present invention, the script further comprises information related to resources required by a third processor; and dynamically allocates the resources at the time needed by the third processor.

[13] According to another aspect of the present invention, a method by a processor for allocating resources for use by one or more tasks in a multi-processor computing environment is taught. The method includes the following: (1) providing a script to the processor, the script containing a map of sequences that will occur during execution of the one or more tasks; (2) parsing the script to determine resources required based on the map of sequences; and (3) allocating the resources immediately prior to execution of the task.

[14] According to another aspect of the present invention, a predictive resource allocation system for a multi-processor computing environment having two or more processors is disclosed. Among other components, the system includes a first processor; a dedicated processor dedicated to providing resource allocation to the first processor; a script file containing information related to the resources required by the second processor; a script engine for running the script file; the dedicated processor in conjunction with the script engine parses the script file to determine the resources required by the second processor; and the dedicated processor dynamically allocating the resources at the time needed by each processor.

[15] According to another aspect of the present invention, the script further comprises information related to resources required by a third processor; and the dedicated processor dynamically allocating the resources at the time needed by the third processor.

[16] According to another aspect of the present invention, a method by a processor for allocating resources for use by two or more tasks in a multi-processor computing environment is disclosed. The method includes: (1) providing a script to the processor, the script containing a map of sequences that will occur during execution of the tasks; (2) parsing the script to determine the map of sequences for the tasks and to determine the resources required by the tasks; and (3) allocating the resources to the tasks such that resource allocation is synchronized with when the resources are needed by tasks.

[17] Advantageously, unlike conventional multi-processor systems wherein all processors may undertake resource allocation, a dedicated processor is solely dedicated to resource allocation and to the task of parsing and interpreting the contents of the script file. Because a script file is provided, multiple processes can be coordinated, and each processor knows which task can be performed without having to wait for the another processor.

BRIEF DESCRIPTION OF THE DRAWINGS

[18] Fig. 1 is a multi-processor environment in which predictive allocation of resources may be employed in accordance with a first embodiment of the present invention.

[19] Fig. 2 is a flow chart of a method employed by a dedicated processor for allocating resources for use by other processors within the multi-processor computing environment of Fig. 1 in accordance with one embodiment of the present invention.

[20] A further understanding of the nature and advantages of the present invention herein may be realized by reference to the remaining portions of the specification and the attached drawings. Further features and advantages of the present invention, as well as the structure and operation of various embodiments of the present invention, are described in detail below with respect to the accompanying drawings. In the drawings, the same reference numbers indicate identical or functionally similar elements.

DETAILED DESCRIPTION OF THE INVENTION

[21] Fig. 1 is a multi-processor environment 100 in which predictive allocation of resources may be implemented in accordance with a first embodiment of the present invention.

[22] As implied by its name, multi-processor environment 100 includes, among other components, one or more processors namely a dedicated processor 102 for controlling a script engine 104; the script engine 104 for processing a script file 106; a first processor 110 for executing two processes namely process 110A and process 116A; and a second processor 114 for executing a process 114A. Processes 110A, 114A and 116A are part of a single application for execution within multi-processor environment 100.

[23] Other components of multi-processor environment 100 includes a memory 108 which is an example of a resource which may be allocated and a communication bus 116. Although not shown, one of ordinary skill in the art will realize that various other resources such as matrix configuration, queue length, priority, etc may be allocated as proves necessary according to the precepts and principles of the present invention.

[24] In operation, a programmer wishing to allocate resources related to the application initially provides a script file 106 to dedicated processor 102. Advantageously, unlike conventional multi-processor systems wherein all processors may undertake resource allocation, dedicated processor 102 is solely dedicated to resource allocation and to the task of parsing and interpreting the contents of script file 106. In one embodiment, script file 106 contains a description of the execution sequence of the application and a list of its corresponding resources, when they are required and when such resources are releasable. On parsing script file 106, dedicated processor 102 becomes privy to the execution sequence and list of resources of the application ahead of its execution. In this manner, the present invention, permits assignment of resources ahead of need and releases them when no longer in use. Although not shown, one of ordinary skill in the art will realize that techniques other than a script file for predictably determining the needs of the multiple processors can be employed. For example, an object-oriented data structure may be provided to dedicated processor 102.

[25] By way of example, an exemplary script file 106 is provided below. In Fig. 1, process 110A and process 116A are executed on processor 110 while process 114A is run by processor 114. Process 116A cannot executed until process 114A is completed. Process 110A requires a 100 Kbyte buffer and process 116A requires a 200 Kbyte buffer. Processor 110A can surrender its 100Kbyte buffer only when process 114A is complete. This allows dedicated processor 102 to reuse the space for processor 116's 200 Kbyte buffer. Script file 106 is as follows:

[26] PROCESS 110A EXECUTES ON PROCESSOR 110; NEEDS: BUFFER 100K REF BB

[27] PROCESS 114A EXECUTES ON PROCESSOR 114

[28] PROCESS 116A EXECUTES ON PROCESSOR 110; WAIT FOR PROCESS 114A; NEEDS BUFFER 200K; FREE BB

[29] Therefore, optimization of memory allocation is decided far in advance of need because dedicated processor 102 possesses a road map for the execution sequence prior to actual execution of the application. It should be observed that actual execution of the application will cause the request for execution of processes 110A, 114A, 116A as well as their requests for buffers. Further, one of ordinary skill in the art will realize that the present invention can be implemented by way of software, hardware e.g. as part of ASIC (application specific integrated circuit) or a combination of both.

[30] According to another embodiment of the present invention, script file 106 is an I/O script having a description of the sequence of I/O operations, resource requirements that will be issued for a given task or thread. For example, the sequence of operations provided for a read sequence may be as follows:

[31] INITIATE READ with — maximum block = 2 KB, source device = 1, destination = 3

[32] READ BLOCK will repeat

[33] CLOSE READ terminates Read sequence

[34] The I/O script does not actually execute the I/O sequence — it informs dedicated processor 102 of future events when the I/O sequence is actually issued so that resources can be assigned when that I/O sequence is activated by the application.

[35] As noted, dedicated processor 102 is solely dedicated to resource allocation and to the task of parsing the contents of script file 106 so it knows in advance the execution sequence of the application. It is for this reason that dedicated processor 102 is able to synchronize exactly when the resources should be allocated. For example, the time to allocate the resource may be 2 milliseconds prior to use by the application. Therefore, in addition to dynamic allocation, the allocation is synchronized for utilization when needed by the other processors. In this manner, the present invention improves overall performance of the multi-processor computing environment.

[36] Fig. 2 is a flow chart of a method 200 employed by dedicated processor 102 for allocating resources to processors 110 and 114 within multi-processor computing environment 100 of Fig. 1 in accordance with one embodiment of the present invention.

[37] In Fig. 2, at step 202, dedicated processor 102 (Fig. 1) is awaiting the receipt of script file 106. Script file 106 contains information related to the resources required by processors 110 and 114.

[38] At block 204, script file 106 is received by dedicated processor 102.

[39] At block 206, after the script file is received, method 200 comprises parsing by dedicated processor 102, script file 106 to determine the resources required by the processors.

[40] At block 208, the resources are dynamically allocated at the time needed by processors 110 and 114. Because script file 106 is provided, multiple processes can be coordinated, and each processor knows which task can be performed without having to wait for the another processor. Dedicated processor 102 knows the what and when certain resources required by processors 110 and 114. For example, if processor 114 requires some memory, it is assigned while processor 110 is busy so that the memory is ready for processor 114 when needed.

[41] In this fashion, the present invention and its embodiments provide a predictive resource allocation system for a multi-processor computing environment. While the above is a complete description of exemplary specific embodiments of the invention, additional embodiments are also possible. Thus, the above description should not be taken as limiting the scope of the invention, which is defined by the appended claims along with their full scope of equivalents.